UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF NEW YORK

CA, INC., D/B/A CA TECHNOLOGIES,

Plaintiff,

v.

NEW RELIC, INC.,

Defendant.

Case No. 2:12-cv-05468-AKT

PLAINTIFF CA, INC.'S OBJECTIONS AND RESPONSE TO SPECIAL MASTER'S REPORT AND RECOMMENDATION

Plaintiff CA, Inc., d/b/a CA Technologies ("CA"), for its objections and response to the Special Master's Report and Recommendation ("Rep.") filed on December 23, 2014 (Dkt. #154), respectfully states as follows.

I. INTRODUCTION

Over a year ago, as part of the Court's claim construction process, the parties agreed to construe "exit code" – a term in every asserted claim of U.S. Patent No. 7,512,935 ("the '935 Patent") – as "a section of object code for execution for multiple normal exits as well as for multiple exceptions." This agreed-upon definition was adopted by this Court in its *Markman* Order. *See* Memorandum & Order (Dkt. # 86) at 3.

Thereafter, the Court appointed Robert Neuner as a Special Master to hear New Relic's motion for partial summary judgment, which is based in part on this claim term. The Special Master reviewed the parties' summary judgment submissions, familiarized himself with the patents and relevant technology, conducted two oral arguments and wrote a detailed Report and Recommendation. While he correctly concluded that the '935 Patent is valid (Rep. at 26) and that the fact finder must determine whether New Relic's .NET Agent infringes the '935 Patent (*id.* at 23-24), his dramatic and unwarranted narrowing of "exit code" in spite of the agreed-upon and Court-ordered definition noted above – and his consequent dramatic and unwarranted narrowing of the '935 Patent itself – led him to incorrectly conclude that New Relic's Java Agent does not infringe the '935 Patent. In sum, stating that "the parties disagree about the import of the definition," the Special Master *sua sponte* reconstrued and narrowed "exit code" long after the close of fact and expert discovery. (*See* Rep. at 26.)

As described below, the new construction recommended by the Special Master is significantly narrower than what was previously specified in the Court's *Markman* order. Aside from being narrower than what the parties agreed to and the Court adopted, it also is substantively incorrect, both because it is inconsistent with the '935 Patent's specification, claims, and prosecution history, and also because it erroneously limits the indefinite article "a" to mean "one and only one," instead of its customary meaning of "one or more." Moreover, the Special Master's recommendation that the claims be narrowed because of a purported disclaimer by the inventor is likewise improper and incorrect because the statements at issue do not, as a matter of law, rise to level of a clear and unambiguous disavowal of claim scope. For these reasons, these aspects of the Special Master's Report should not be adopted.

Additionally, the Special Master's redefinition of a claim term that contravenes the parties' agreement, long after the claim construction phase of the case, is procedurally objectionable. As a result, CA objects for the additional reason that the Special Master exceeded the authority conferred to him by the Court's Order of Appointment of and Reference to Special Master (Dkt. #151), dated September 12, 2014. In short, the Special Master was assigned to recommend whether New Relic should be entitled to summary judgment based upon the patent claims and terms as they had already been construed by the parties and the Court. He was not assigned to re-construe key terms and thereby narrow the scope of the ’935 Patent. If New Relic’s software agents operate in such a way that they would infringe the ’935 Patent under the agreed-upon and Court-adopted construction of “exit code,” then CA is entitled to have the fact-finder determine whether those software agents infringe the ’935 Patent.

CA therefore respectfully requests that the Court reject the Special Master’s reinterpretation of “exit code” and narrowing of the ‘935 Patent, and deny New Relic’s motion for partial summary judgment on non-infringement in all respects because the recommendation to grant those aspects of the motion is based upon those errors.

II. <u>THE ’935 PATENT</u>

Performance profiling tools are software products that monitor the health of other software applications. For example, Amazon.com might use such a tool so that it can determine whether there are application performance problems on its website that are causing its customers to experience delays during their shopping experience. Among other metrics, performance profiling tools might measure the amount of time it takes a software routine[1] (for example, a routine that allows one to view the contents of a virtual shopping cart) to execute. (’935 Patent, col. 1, ll. 1-11.) Although the ’935 Patent is not limited to performance profiling, the ’935 inventions are advantageously used in a performance profiling system, such as the accused New Relic agents.

When a performance profiling tool is installed, it is able to tap into the routines of the monitored software through a process known as “instrumentation.” A method is “instrumented” by adding special instructions to the targeted application’s code, such as adding code to a method

[1] A routine is a named section of computer code that performs one or more tasks, and is also known as a function, procedure, subroutine, or method. (’935 Patent, col. 3, ll. 46-49.)

to start a timing mechanism when the method is called (*i.e.*, starts), and to stop the timing mechanism when the method exits (*i.e.*, ends), thereby generating the desired timing data.

One of the key issues addressed by the ’935 Patent is ensuring that the profiling process (in the example above, the timing mechanism) is stopped regardless of how the method exits. There are two types of exits from a method: normal or exception. A normal exit is an intended exit when no errors occur within the method. For example, having an item appear in an online shopping cart would indicate a normal exit for the routine associated with the “add to basket” method. An exception, on the other hand, causes an unintended exit arising from an error, *e.g.*, the web application is unable to access the database and thus cannot add the item.

Object code is computer code that is “machine executable or suitable for processing to produce executable machine code,” that is, code that can be executed by a processor. (*Id.*, col. 1, ll. 24-25.) The ’935 Patent discloses and claims various techniques to instrument a method by “adding new instructions to the object code and/or modifying existing instructions in the object code.” (*Id.*, col. 1, ll. 28-30.) In particular, the ’935 Patent teaches “a system for adding performance profiling functionality to object code such that the new functionality is provided for all (or almost all) exits of the method (or other set of code).” (*Id.*, col. 2, ll. 9-12.) Figure 1, below, depicts a method comprised of two sections of object code, and their respective normal exits, prior to instrumentation.

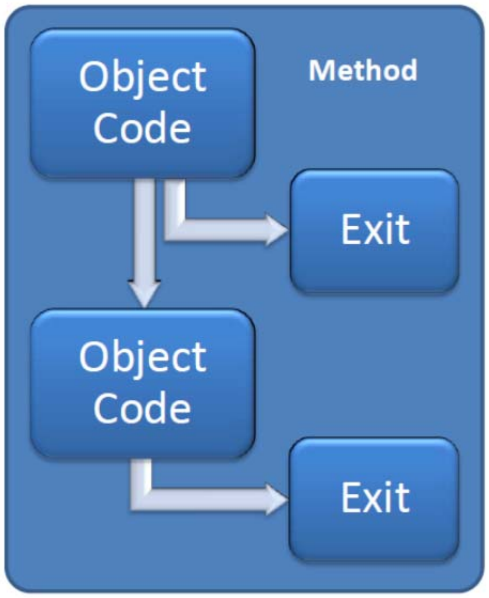

Figure 1.

As discussed in the Background of the Invention, while prior art systems added performance profiling functionality at normal exits (*id*., col. 1, ll. 55-57), the prior art failed to account for exceptions:

> Adding performance profiling code to each of the instructions that each can be an exit has its drawbacks. First, the same code may be added in several places which can significantly increase the amount of code in an application. A significant increase in code may impact the performance of the application and increase the chance of adding an error to the application. Additionally, the greater the number of exits, the greater chance that the system adding performance profiling code will miss one of the exits. Finally, <u>not all of the exits are explicitly stated in the code</u>. For example, <u>there can be errors or exceptions</u>. When an error or an exception occurs, the normal flow of the method can be halted, in which case <u>none of the explicit exits will be performed</u> and an exit associated with the error or exception will be performed. Previous performance profiling tools have not adequately accounted for all the possible exits in the software.

(*Id*., col. 1, l. 60 – col. 2, l. 8, emphasis added.) According to the ’935 Patent, “what is needed is a system for adding performance profiling functionality to object code such that the new functionality is provided for all (or almost all) exits of the method (or other set of code).” (*Id*., col. 2, ll. 9-12.) These passages plainly illustrate that the inventions taught and disclosed in the ’935 Patent are distinguished from the prior art by handling an “exit associated with the error or exception” <u>in addition to</u> the normal exits. The ’935 specification, claims, and prosecution history each support this interpretation.

The Summary of the Invention, which is part of the specification, demonstrates that the ’935 Patent discloses multiple embodiments in which “code is added to the existing object code such that the relevant action is performed at each of the exits” to address the shortcomings of the prior art. (*Id.*, col. 2, ll. 24-25.) The ’935 specification then clearly and explicitly identifies distinct embodiments, for example:

> In <u>one implementation</u>, the method of the present invention comprises the steps of adding exit code to the existing object code and adding an entry to an exceptions data store pointing to the exit code. <u>One alternative</u> also includes adding start code to the existing object code; however, some embodiments do not add start code. <u>The start code, the exit code and the entry into the exceptions data store are used to add the additional functionality</u>. The additional functionality can be any functionality desired to be added to the existing object code. One example is to add performance profiling functionality.

(*Id.*, col. 2, ll. 36-44, emphasis added.) This embodiment is claimed in each of the asserted independent claims, which require “adding a new entry in an exception data store” to add an exception handler “for multiple types of exceptions.” (*Id.* at col. 16, ll. 40-49 (claim 1); col. 18, ll. 16-26 (claim 22); col. 19, ll. 26-34 (claim 35).) This embodiment is shown conceptually in the following illustration, in which the “try” and “catch” blocks (in lime green) are added to the original method by virtue of “adding a new entry in an exception data store,” thus creating the exception handler, and the exit code added to the original method (in green):

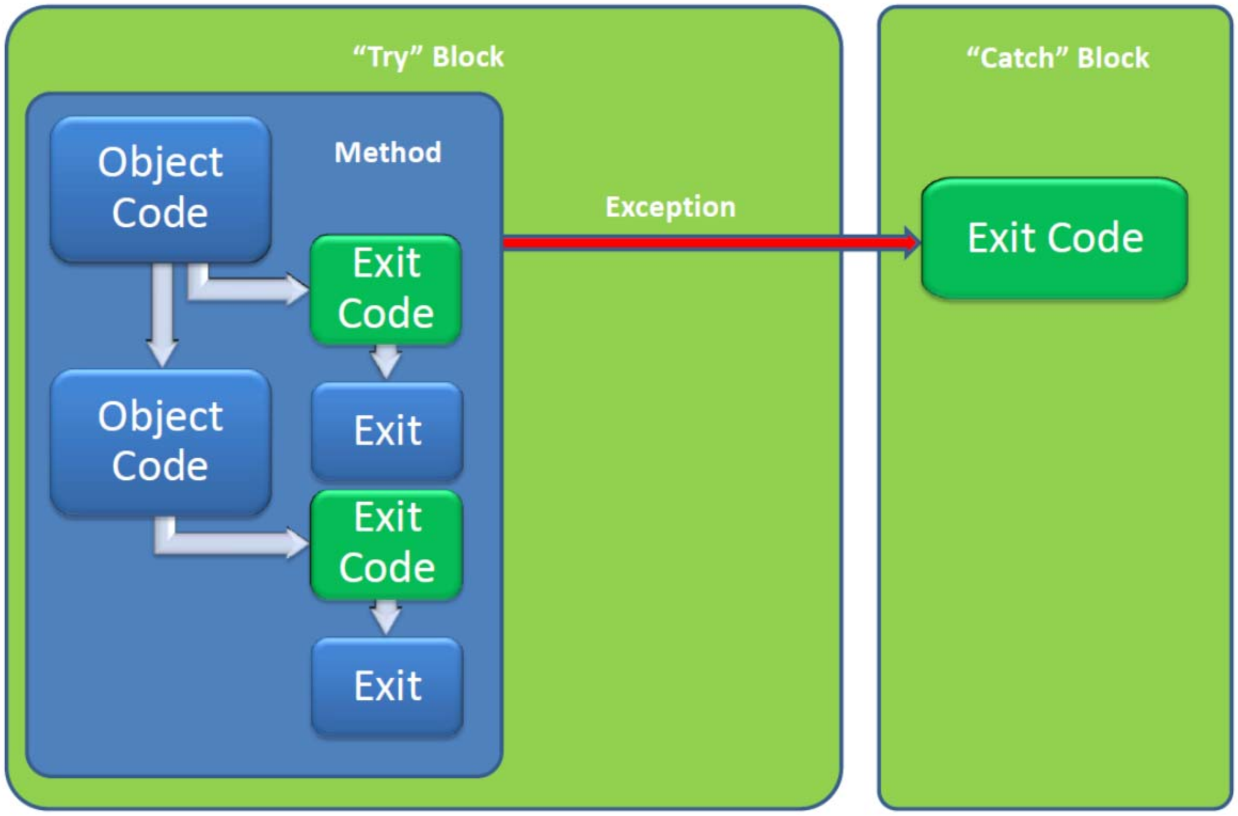

Figure 2.

In the Figure 2 embodiment above, using a technique known as "in-lining," the same block of exit code is added immediately before each of the two normal exits and will be executed prior to the normal exit. With respect to exceptions, because the original method is now within the added "try" block, if an exception occurs in the method, the exit code in the "catch" block will be executed.

A narrower embodiment of the '935 Patent teaches the addition of a "jump" instruction to the existing object code:

> In <u>one embodiment</u>, the start code starts an action and the exit code stops the action. The start code is positioned to be executed previous to the original byte code and the exit code is positioned to be executed subsequent to the original byte code. The step of adding exit code includes adding an instruction to jump to the exit code from the original byte code.

(*Id.*, col. 2, ll. 46-51, emphasis added.) A jump instruction is a computer instruction that transfers execution to a different instruction in the object code, *i.e.*, it "jumps," to a different instruction in the object code. Several dependent claims are drawn to this particular embodiment. For example, claim 4 recites "[a] method according to claim 3, wherein said creating a path includes adding a jump instruction." (*Id.*, col. 16, ll. 56-57.) This embodiment is shown conceptually in the following illustration:

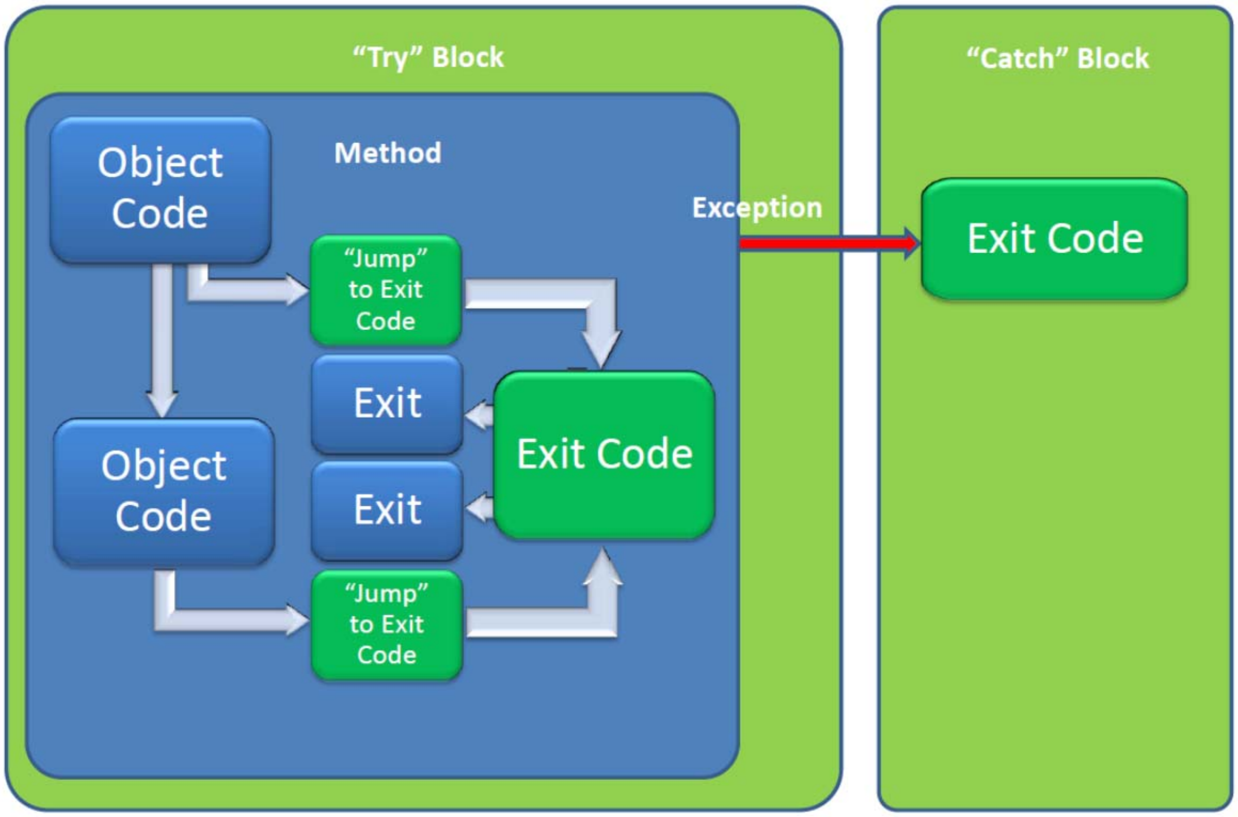

Figure 3.

In the Figure 3 embodiment depicted above, a "jump" instruction will execute before each normal exit, so that any flow directed to a normal exit will first jump to the block of added exit code. Once that exit code has been executed, flow returns to the respective normal exit. Like the Figure 2 embodiment discussed above, if an exception occurs in the method, the exit code in the "catch" block will be executed.

The narrowest embodiment of the ’935 Patent adds “try and finally functionality” to the existing object code:

> In <u>one embodiment</u> using Java byte code, the byte code for a method and the method’s exception table can be modified to implement the functions of the source code statements “try” and “finally.”

(*Id*., col. 2, ll. 32-35, emphasis added.) “Try and finally functionality,” construed by the parties as “the execution of a block of code (‘finally’) regardless how an associated block of code (‘try’) exits” is a programming construct that ensures that code in the “finally” block is executed upon any exit from the “try” block, where a “block” is a collection of object code instructions. This is the narrowest of the embodiments because it provides for only one result, *i.e*., regardless of the method outcome in the try block, the exit code in the finally block is executed. Several dependent claims are drawn to this embodiment. For example, claim 11 recites “[a] method according to claim 5, wherein: said adding exit code and adding a new entry include adding try and finally functionality to said original byte code.” (*Id*., col. 17, ll. 20-23.) The “try and finally functionality” embodiment is shown conceptually in the following illustration:

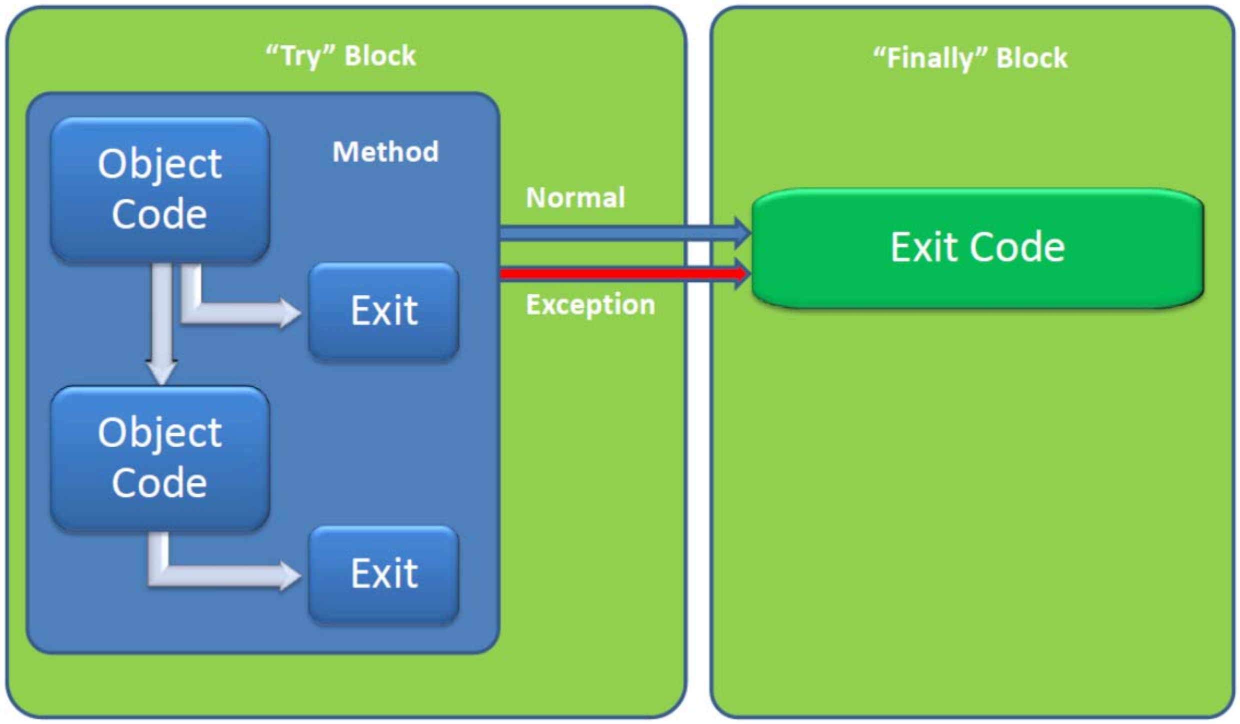

Figure 4.

In the Figure 4 embodiment depicted above, regardless of the type of exit occurring within the try block, *i.e.*, a normal exit or exception, the code in the finally block will execute.

III. <u>THE SPECIAL MASTER IMPROPERLY NARROWED THE CONSTRUCTION OF "EXIT CODE"</u>

CA objects to the Report because the Special Master narrowed the parties' agreed-upon and Court-ordered construction for "exit code" such that it now reads on only the narrowest embodiment of the '935 Patent (Figure 4 above) and, based on that new and narrowed construction, recommended granting New Relic's motion in part. As re-construed by the Special Master, "exit code" now excludes all of the other, broader inventions (Figures 2 and 3 above) – including those that fall within the scope of the agreed-upon construction of "exit code." This approach is clearly erroneous. It is well-settled Federal Circuit law that the limitations associated with various embodiments, such as those of the ’935 Patent described in the specification (including try/finally functionality), cannot be imported into the claims:

> [A]lthough the specification often describes very specific embodiments of the invention, we have repeatedly warned against confining the claims to those embodiments . . . . In particular, we have expressly rejected the contention that [even] if a patent describes only a single embodiment, the claims of the patent must be construed as being limited to that embodiment. That is not just because section 112 of the Patent Act requires that the claims themselves set forth the limits of the patent grant, but also because persons of ordinary skill in the art rarely would confine their definitions of terms to the exact representations depicted in the embodiments.

Phillips v. AWH Corp., 415 F.3d 1303, 1323 (Fed. Cir. 2005) (en banc) (internal quotations and citations omitted); *id*. (noting that because the claim language, not the specification, describes the scope of the patented invention, the specification may not alter the scope of the claim, and the court must not import limitations in the specification not found in the claim language); *see Innova/Pure Water, Inc. v. Safari Water Filtration Sys., Inc.*, 381 F.3d 1115, 1116-17 (Fed. Cir. 2004) (“[T]he claims of the patent, not its specifications, measure the invention . . . . Accordingly, particular embodiments appearing in the written description will not be used to limit claim language that has broader effect.”)

Indeed, the Special Master tacitly acknowledges that, as the parties had construed “exit code,” it encompasses “adding the same performance profiling code to each of the instructions that can be an exit.” (Rep. at 17; Figure 1.)[2] Yet, the Special Master disregarded the parties’ carefully negotiated construction of “exit code” and substituted his own interpretation, one even narrower than the interpretation that New Relic agreed to during claim construction. The Special

[2] This is precisely what CA had argued, *i.e.*, that “a section of object code” can mean one block of code replicated and inserted at various places. However, and notwithstanding this acknowledgment, the Special Master’s recommendation denies CA the full scope of the agreed-upon construction.

Master's erroneously narrow construction of this key claim term infects all of his recommendations regarding infringement. For this reason, the Special Master's construction of "exit code" and resulting recommendations should not be adopted by the Court.

A. The Special Master Misstated the Scope of the '935 Claims

The error in the Report and Recommendation begins with a fundamental misunderstanding of the teaching of the '935 Patent and the scope of its claims. The '935 specification acknowledges that adding exit code at each normal exit of a method—without more—was known in the prior art. The invention of the '935 Patent, on the other hand, ensures that the exit code is executed regardless of how the method exits, *i.e.*, normally or if an exception occurs. As discussed above, the '935 Patent teaches adding code both at normal exits and adding an exception handler to the existing object code of the instrumented method ("adding a new entry in an exception data store") to handle exception exits.

The specification, meanwhile, teaches <u>at least two ways</u> to handle normal exits: (1) adding exit code to the existing object code at each normal exit (Figure 2); and (2) placing exit code in a single block that is added to the existing object code, which will be jumped to before both normal and exception exits (Figure 4). (*Id.*, col. 2, ll. 19-32; col. 5, ll. 5-64.) None of the prior art cited in the Report and Recommendation – the discussion in the Background of the Invention, the Berry references, and Introscope 1.0 – added both code at normal exits *and* an exception handler to the existing object code to execute exit code for exception exits. Indeed, none of this prior art teaches adding an exception handler to existing object code – an essential element of all '935 claims – for any purpose.

The Special Master states CA's position on the scope of the '935 independent claims:

> The conclusion that CA draws from its argument is that the independent claims of the '935 Patent cover and thus are infringed

> by any process which includes as a first step the prior art technique of adding exit code to the existing object code at each normal exit and as a second step the novel technique of adding an exception handler containing exit code for exception exits.

(Rep. at 18.) CA has not parsed the ’935 claims in this manner. CA contends that adding an exception handler to existing object code – performed by each of the accused New Relic agents – was not disclosed in the prior art discussed in the Report and Recommendation. All of the prior art relied upon by the Special Master was considered by the Examiner during prosecution, and the claims were allowed over these references. The applicant, Mr. Cobb, successfully distinguished the independent claims on the basis that none of the cited prior art taught or suggested the addition of an exception handler to existing object code as a way to ensure that exit code was executed for exception-caused exits. (*See, e.g.*, Dkt. #115, Decl. of Corey Johanningmeier, Exhibit L, First Cobb Declaration at NEWRELIC 403738, ¶ 17; Cobb Second Declaration at NEWRELIC 403779-80, ¶¶ 16-17.)

The fundamental misunderstanding of the scope of the ’935 Patent is further demonstrated by the Special Master’s characterization of the ’935 Patent as containing only a single embodiment (Figure 4), which cannot be squared with the other explicitly disclosed (but ignored) embodiments (Figures 2 and 3). (*See* ’935 Patent, col. 2, ll. 26-32; col. 2, ll. 36-44; col. 2, ll. 46-51.) The Special Master characterized the ’935 Patent as follows:

> In **<u>the</u>** embodiment disclosed in the ’935 patent specification, rather than physically insert copies of the exit code at every explicit exit and different code or no code at exception exits, the method of the ’935 patent conceptually encloses the byte code within a “try” block and places the “exit code” within a “finally” block. By doing so, the same block of code – that is the same instructions at the same location in memory – is reached by all of the exits, both normal and exception.

(Rep. at 3, emphasis added.) Thus, according to the Special Master, only the embodiment shown in Figure 4 was disclosed.

This central flaw undermines the Special Master's infringement analysis. While the dependent claims drawn to the "try and finally functionality," *e.g.*, claim 11, do indeed find support in the "try / finally" implementation, the remaining independent and dependent claims find explicit support in the other embodiments disclosed in the specification. Moreover, the Special Master criticizes the parties' agreed construction for "exit code" because it is "more expansive than the disclosed try / finally implementation wherein the same block of code is reached by all of the exits, both normal and exception." (Rep. at 17.) What the Special Master did not appreciate is that the parties crafted this construction precisely because there are multiple embodiments of varying claim scope, requiring a commensurately broad definition for "exit code."[3]

Similarly, the Special Master found that "a section" as used in the parties' agreed construction means "one and only one section," despite the repeated teaching in the '935 specification – including through the ignored embodiments – supporting a broader meaning for "a" of "one or more." (Rep. at 23.) The Special Master then used this finding to justify excising from the '935 Patent the broader embodiments because they do not utilize a single, solitary section of exit code. The Federal Circuit, however, has held that "a" means "one or more" as a matter of law, particularly as here, where the claims use the open phrase "comprising." *KCJ Corp. v. Kinetic Concepts, Inc.*, 223 F.3d 1351 (Fed. Cir. 2000). Indeed, the Special Master elsewhere in the Recommendation acknowledges that "a section" does not necessarily mean a

[3] Naturally, as this narrower construction now suits New Relic's non-infringement theory, it does not object to having its prior agreement on claim construction usurped.

single, solitary section of exit code. The Special Master acknowledges that the term encompasses "adding the same performance profiling code to each of the instructions that can be an exit." (Rep. at 17). This is precisely the embodiment demonstrated in Figure 2.

B. The Special Master's Construction of "Exit Code" Violates the Doctrine of Claim Differentiation

The effect of the Special Master's construction is to read the "try / finally" limitation into every claim of the '935 Patent, which violates the doctrine of claim differentiation. Under this doctrine, independent claim 1 *must* be broader than the "try and finally functionality" of dependent claim 11. "Where [] the sole difference between the independent claim and the dependent claims is the limitation that one party is trying to read into the independent claim, 'the doctrine of claim differentiation is at its strongest.'" *SanDisk Corp. v. Kingston Tech. Co., Inc.*, 695 F.3d 1348, 1361 (Fed. Cir. 2012) (quoting *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 910 (Fed. Cir. 2004), citing *Phillips v. AWH Corp.*, 415 F.3d 1303, 1315 ("[T]he presence of a dependent claim that adds a particular limitation gives rise to a presumption that the limitation in question is not in the independent claim.")).

C. The Applicant Did Not Disclaim The Broader Embodiments of the '935 Patent

The Special Master's error began by disregarding all but the narrowest embodiment in the '935 specification, and continued by his also incorrectly finding that statements made by the inventor during prosecution of the '935 Patent constituted a disclaimer of the other embodiments set forth in the '935 Patent. In other words, the Special Master found that the '935 inventor, Mr. Jeffrey Cobb, disclaimed all embodiments other than "try and finally," *i.e.*, disclaimed the embodiments shown in Figures 2 and 3 above, despite the fact that (i) those embodiments are plainly disclosed and claimed in the '935 Patent, (ii) the language asserted to be the disclaimer was not directed to these other claimed embodiments, and (iii) at best, the language claimed to be the disclaimer is subject to multiple, reasonable interpretations meaning that, as a matter of law, it does not operate as a disclaimer.

Any disclaimer analysis begins with a "'heavy presumption' that claim terms carry their full ordinary and customary meaning, unless the patentee unequivocally imparted a novel meaning to those terms or expressly relinquished claim scope during prosecution." *Omega Eng'g, Inc. v. Raytek Corp.*, 334 F.3d 1314, 1323 (Fed. Cir. 2003) (emphasis supplied) (citing *CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1366 (Fed. Cir. 2002)); *Home Diagnostics, Inc. v. LifeScan, Inc.*, 381 F.3d 1352, 1358 (Fed. Cir. 2004) ("Absent a clear disavowal in the specification or the prosecution history, the patentee is entitled to the full scope of its claim language.") (emphasis supplied).

Mr. Cobb's statements distinguishing several prior art references during prosecution do not constitute disclaimer. The Special Master cites several passages from Mr. Cobb's two declarations during the '935 prosecution in order to show that Mr. Cobb distinguished the prior art on the ground that they did not disclose "try / finally," but omits the critical fact that Mr. Cobb was at the time distinguishing the dependent claims (*e.g.*, claim 11) that explicitly reference "try and finally functionality." (*See*, *e.g.*, Dkt. #115, Decl. of Corey Johanningmeier, Exhibit L, First Cobb Declaration at NEWRELIC 403740, ¶ 21.) Thus, the purported disclaimer language that distinguished the '935 Patent from the prior art with reference to "try and finally functionality" (Figure 4) was not part of the inventor's discussion of the independent claims that disclose the other, broader embodiments (Figures 2 and 3). Far from disclaiming the other broader embodiments that are disclosed and claimed in the '935 Patent, the quoted passages demonstrate merely that the inventor painstakingly distinguished the prior art cited by the Examiner based on the express limitations in each of the claims discussed. This misperception and misapplication of the purportedly disclaiming statements leads to a fundamental error in the Special Master's analysis.

Indeed, with respect to the broadest claims, *e.g.*, independent claim 1, Mr. Cobb distinguished the prior art without mention of "try / finally" at all. In his First Declaration (Dkt. #115, Decl. of Corey Johanningmeier, Exhibit L – excerpts of '935 Prosecution History at NEWRELIC 403735-42), Mr. Cobb distinguished one of the Berry references cited by the Examiner (U.S. Patent No. 6,728,955) on a limitation-by-limitation basis. For example, using the language from pending claim 1, Mr. Cobb stated "there is nothing in Berry that would teach or even suggest <u>adding a new entry in an exceptions data store for existing object code where the new entry points to exit code added to the existing object code</u>." (*Id*. at NEWRELIC 403738, ¶ 17, emphasis added.) As a further example, in his Second Declaration (*id.* at NEWRELIC 403777-87), Mr. Cobb stated "Introscope 1.0 does not include, or otherwise embody, functionality for 'adding a new entry in an exceptions data store for said existing object code, said new entry points to said exit code," as recited in claims 1 and 82, . . . Introscope 1.0 does not add entries to exceptions data stores." (*Id*. at NEWRELIC 403779, ¶ 16.) Thus, in each of his two declarations, Mr. Cobb carefully explained that the limitations in each claim were not taught or disclosed in those prior art references. This salient fact is not recognized or accounted for in the Report, further revealing the error.

As such, no clear and unmistakable disclaimer exists. At best, these statements from the '935 Patent's prosecution history are subject to multiple reasonable interpretations and cannot, as a matter of law, form the basis of disclaimed subject matter. *See 3M Innovative Properties Co. v. Tredegar Corp.*, 725 F.3d 1315, 1326 (Fed. Cir. 2013) ("Where an applicant's statements are amenable to multiple reasonable interpretations, they cannot be deemed clear and unmistakable."). Therefore, the Report's reliance upon these statements as demonstrating disclaimer is unfounded and therefore improper.

IV. THE ACCUSED NEW RELIC AGENTS INFRINGE THE ASSERTED CLAIMS

When compared to a properly-construed '935 Patent (one that discloses all the embodiments), it is clear that New Relic's agents infringe. As the Special Master notes, the parties do not dispute the operation of New Relic's accused Java agent and .NET agent. As correctly stated by the Special Master, the New Relic Java agent adds exit code at each of the normal exits of an instrumented method and adds an exception handler containing exit code for exception exits of the instrumented method. (Rep. at 12.) This implementation is essentially the embodiment shown above in Figure 2. The Java agent literally meets the parties' agreed construction of "exit code," and therefore clearly infringes that limitation of the asserted claims. Accordingly, New Relic's motion for partial summary judgment with respect to the Java agent should be denied.

The Special Master's correctly describes how the two versions of the .NET agent operate. The original version of the .NET agent adds jump instructions at each normal exit, causing a jump to one section of exit code that is executed before exiting the method. An exception handler is added to the method by the .NET agent. A different section of exit code in this exception handler is executed if an exception occurs. This implementation is essentially the embodiment shown above in Figure 3. The original version thus meets the parties' agreed construction for "exit code" literally, and this limitation is also clearly infringed.

In the later version of the .NET agent, beginning with version 2.9.135.0, the original method is replaced by a new method, and an exception handler is added. The new method contains the start code and exit code, and calls the original method. After the original method is called, one section of exit code is executed if the original method exited normally, and another section of exit code, within the added exception handler, is executed if the original method exited because of an exception. Although this variant of the .NET agent does not use jump instructions, it is conceptually like Figure 3, shown above. As such, the later versions of the .NET agent also literally meet the "exit code" limitation, and therefore infringe.

Because each of the accused versions of the Java agent and .NET agent add an exception handler containing exit code for exception exits – the accused agents not only read literally on the asserted claims of the '935 Patent, they are also distinct from the prior art relied upon by the Special Master. Accordingly, the Court should reject the Special Master's recommendation to grant summary judgment of non-infringement as to any of the accused agents.

V. INVALIDITY

The Special Master identified seven discrete issues of material fact "whose resolution is crucial to New Relic's defense of anticipation" and correctly decided that they are for the jury to decide at trial. (Rep. at 25.) Accordingly, the Special Master recommended that the Court deny New Relic's motion for summary judgment that all of the asserted claims are invalid as anticipated by the Dahm Article and the JavaClass system, which CA endorses.

VI. CONCLUSION

CA respectfully requests that the Court adopt the Special Master's recommendation to deny New Relic's motion for partial summary judgment of invalidity, and further requests that the Court reject the Special Master's narrowed construction for "exit code" and deny New Relic's motion for partial summary judgment of non-infringement.

Date: January 13, 2015

<u>/s/ Barry K. Shelton</u>
Barry K. Shelton
Conor M. Civins
Matthew K. Gates
111 Congress Avenue, Suite 2300
Austin, Texas 78701-4061
Telephone: (512) 472-7800
Facsimile: (800) 404-3970
Email: barry.shelton@bgllp.com
Email: conor.civins@bgllp.com
Email: matt.gates@bgllp.com

David J. Ball
1251 Avenue of the Americas
New York, New York 10020
Telephone: (212) 508-6100
Facsimile: (800) 404-3970
Email: david.ball@bgllp.com

Attorneys for Plaintiff CA, Inc., d/b/a CA Technologies

CERTIFICATE OF SERVICE

I hereby certify that on January 13, 2015, I electronically filed the foregoing with the Clerk of the Court using the CM/ECF system, per Local Rule CV-5(a)(3), which will send notification of such filing via electronic mail to all counsel of record. All counsel of record were served with the sealed filing via electronic mail.

/s/ Barry K. Shelton
Barry K. Shelton